SORTING ALGORITHM VISUALIZER

UCS503 Software Engineering Project Report

End-Semester Evaluation

Submitted by:

(<102003702>) Ganesh Mittal

(<102003704>) Vishakha

(<102003710>) Neeraj Sandal

BE Third Year, COE

Group No: 3COE28

Submitted to:

Mrs. Anamika Sharma

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

TIET, Patiala

December 2022

TABLE OF CONTENTS

1.1 Software Bid

Group : 4 Dated: 08-08-2022

Team Name: Hustlers

Team ID (will be assigned by Instructor):

Please enter the names of your Preferred Team Members. :

- You are required to form **a three to four person** teams'
- Choose your team members wisely. You will not be allowed to change teams.

Name	Roll No	Project Experience	Programming Language used	Signature
Ganesh Mittal	102003702	Mess management System	SQL, MySQL, HTML,CSS	[illegible]
Vishakha	102003704	Airport Management System	SQL, MySQL, HTML,Javascript	Vishakha
Neeraj Sandal	102003710	Android app	HTML,CSS,DSA	Neeraj

Programming Language / Environment Experience

List the languages you are most comfortable developing in, **as a team**, in your order of preference. Many of the projects involve Java or C/C++ programming.

1. C++
2. HTML
3. JavaScript
4. Bootstrap

Choices of Projects:

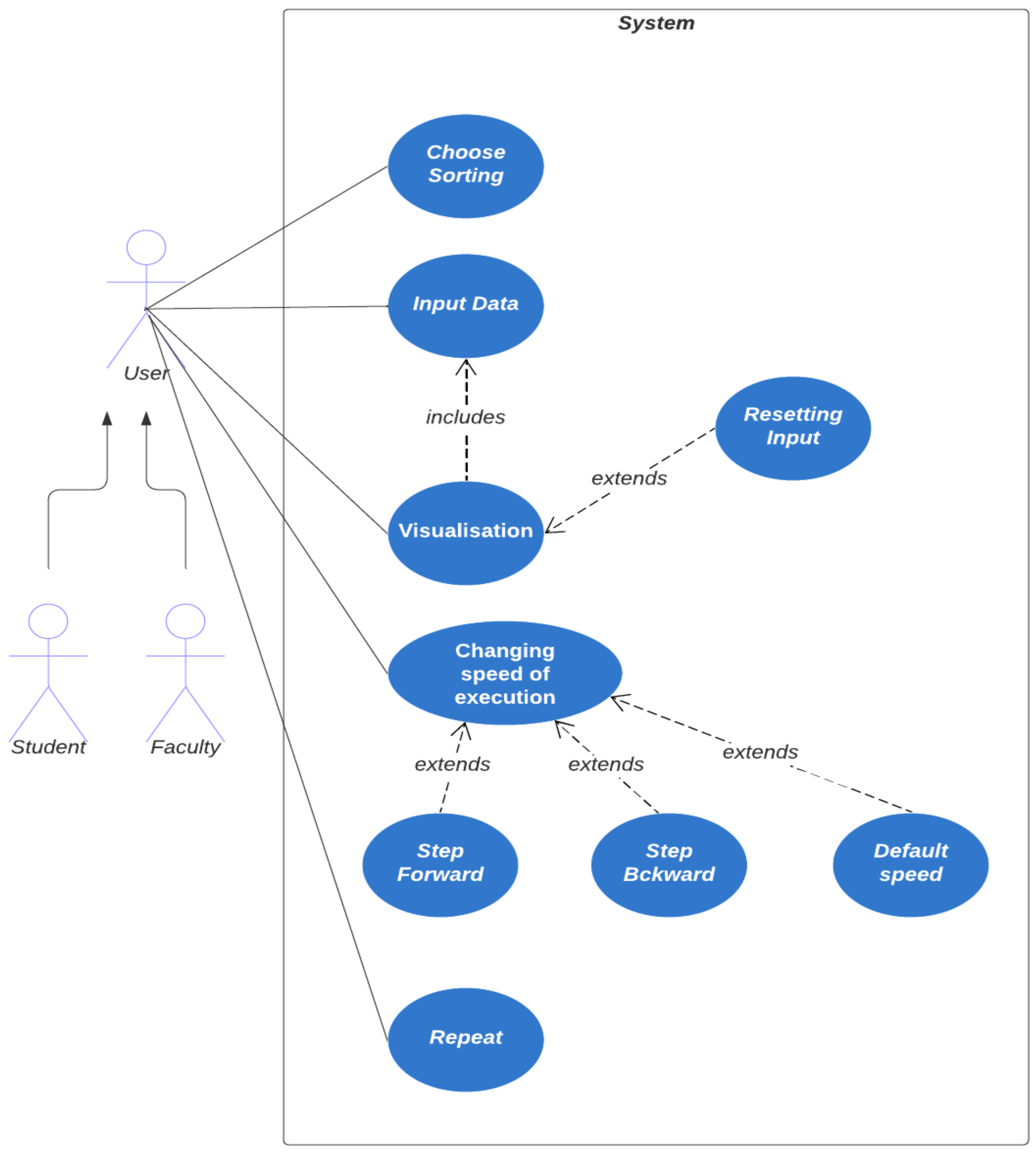
Please select **four projects** your team would like to work on, by order of preference: [Write at least one paragraph for each choice (motivation, the reason for the choice, feasibility analysis, etc.)]

First Choice	Sorting Algorithm Visualizer An animated way to explain the sorting algorithms in an interactive way.
Second Choice	Learning Management System (LMS) A software application or web-based technology is used to plan, implement and assess a specific learning process.
Third Choice	Fingerprint Scanner A scanner system that can be used for attendance system by scanning fingerprints.
Fourth Choice	Khatta Book App An app to create bills online for small shopkeepers.

1.2 Project Overview

This project discusses a study performed on animating sorting algorithms as a learning aid for classroom instruction. The web-application animation tool was created to visualize six common sorting algorithms: Selection Sort, Bubble Sort, Insertion Sort, Heap Sort, Quick Sort, and Merge Sort. The animation tool would represent data as a bar graph and dot plot. After selecting a data ordering algorithm, the user can run an automated animation or step through it at their own pace.

2.1.1 Use Case Diagram



2.1.2 Use Case Scenarios

Use case Scenario 1

1. Use Case Title	Choose Sorting
2. Abbreviated Title	Choose Sorting
3. Use Case Id	1
4. Actors	User

5. Description:

It allows user to select particular type of sorting from the given list.

5.1 Pre-Conditions:

5.2 Task Sequence:

1. User enters the website and selects the type of sorting.

2. After selection of particular sort, information related to that sorting will be provided.

3. Pseudocode , time complexity and space complexity will displayed.

5.3 Post Condition:

User inputs the data and then executes the algorithm.

6. Modification History: 10-Oct-2022

7. Author: Ganesh , Neeraj, Vishakha

Use case Scenario 2

1. Use Case Title	Visualisation
2. Abbreviated Title	Visualisation
3. Use Case Id	2
4. Actors	User

5. Description:

It allows user to understand the working of its selected sorting algorithm i.e. its step by step execution in a statistical way.

5.1 Pre-Conditions:

Selection of sorting with considerable size of input data.

5.2 Task Sequence:

After choosing a particular sort with input data, user is directed for visual execution of that algorithm.

5.3 Post Condition:

During visualization, user sets its execution speed accordingly with the feature of “step forward” and “step backward”.

6. Modification History: 14-Oct-2022

7. Author: Ganesh , Neeraj, Vishakha

Use case Scenario 3

1. Use Case Title	Execution Speed
2. Abbreviated Title	Exe. speed
3. Use Case Id	3
4. Actors	User

5. Description:

Here the user manages the execution speed accordingly(default,step forward,step backward)

5.1 Pre-Conditions:

User must execute particular sorting algorithm.

5.2 Task Sequence:

At the time of execution, user can modify the execution speed.

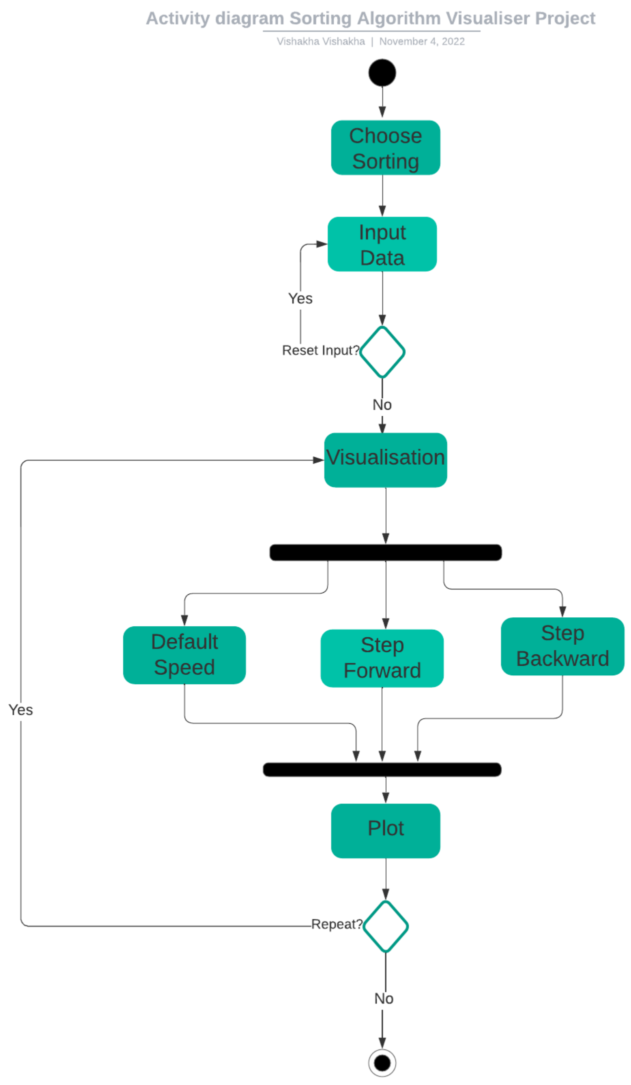
5.3 Post Condition:

User can repeat the whole execution.

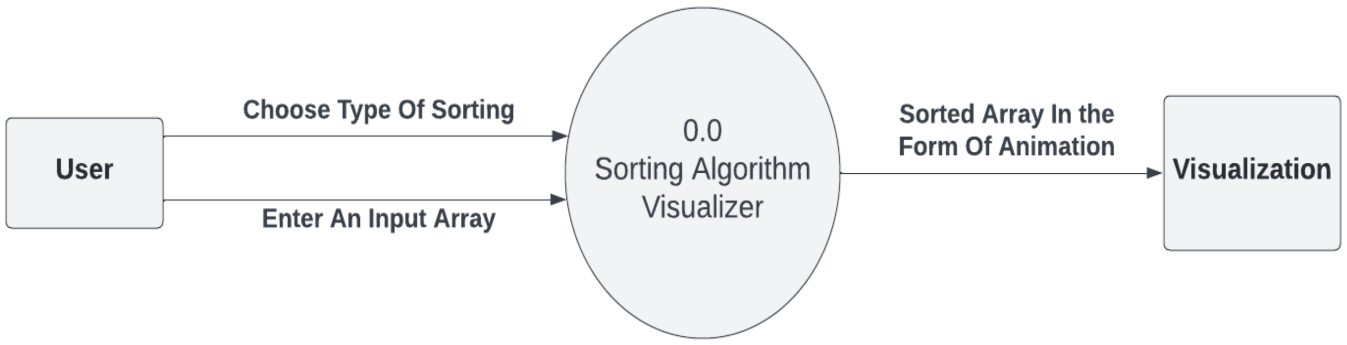
6. Modification History: 19-Oct-2022

7. Author: Ganesh , Neeraj, Vishakha

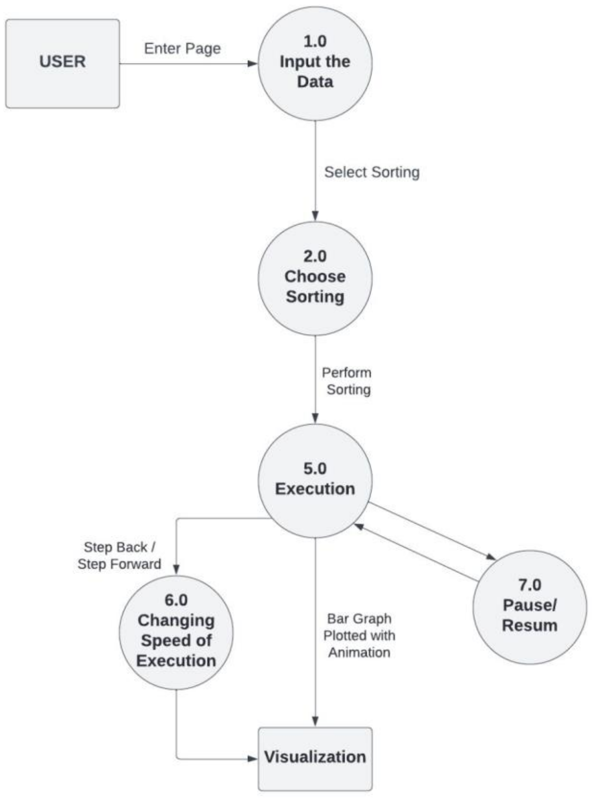
2.2 Activity Diagram



2.3.1 DFD Level-0



2.3.2 DFD Level-1



2.4 Software Requirement Specification in IEEE Format

Software Requirements Specification

for

<Sorting Algorithm Visualiser Project >

Version 1.0 approved

Prepared by :

Ganesh Mittal<102003702>
Vishakha<102003704>
Neeraj Sandal<102003710>

<Thapar Institute of Engineering and Technology, Patiala>

<Date:12-09-2022 >

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

2. Introduction

Purpose

This Project is based on the visualization of the algorithms (Selection Sort, Insertion Sort, Bubble Sort). It creates a teaching support software. This application supports a graphic visualization of selected algorithms on randomly generated or manually created arrays, step-by-step execution possibilities, and the current state of variables.

Definitions, Acronyms, and Abbreviations

Definitions

Table 1 explains the most commonly used terms in this SRS document.

Sr. No.	**Term**	**Definitions**
1.	Algorithm	An algorithm is a set of instructions for solving a problem or accomplishing a task. An example of an algorithm is a recipe, which consists of specific instructions for preparing a dish or meal. Every computerized device uses algorithms to perform its functions in the form of hardware- or software-based routines.
2.	Sorting	Sorting refers to arranging data in a particular format. The sorting algorithm specifies the way to put data in a specific order. Most standard charges are in numerical or lexicographical order.
3.	Complexity	The complexity of an algorithm is a function describing the algorithm’s efficiency in terms of the amount of data the algorithm must process. Usually, there are natural units for the domain and range of this function. There are two main complexity measures of the efficiency of an algorithm: • Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of

		input to the algorithm. • Space complexity is a function describing the amount of extra memory (space) an algorithm takes regarding the amount of input to the algorithm.
4.	Traverse	To traverse an array means to access each element (item) stored in the collection so that the data can be checked or used as part of a process.
5.	Swap	Swapping is a valuable technique that enables a computer to execute programs and manipulate data files more significantly than the main memory.
6.	Pseudo Code	Pseudocode is a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than a programming language.
7.	Dry Run	A dry run is the process of a programmer manually working through their code to trace the value of variables. There is no software involved in this process. Traditionally, a dry run would involve a printout of the code.
8.	Array	An array is a linear data structure that collects elements of the same data type and stores them in contiguous and adjacent memory locations

List of Abbreviations

Table 2 gives the complete form of the most commonly used mnemonics in this SRS document.

Sr. No.	Mnemonic	Full Forms
1.	JS	JavaScript
2.	URL	Uniform Resource Locator
3.	HTML	Hypertext Markup Language
4.	CSS	Cascading Style Sheets
5.	DOM	Document Object Model
6.	UI	User Interface
7.	UX	User Experience
8.	SDLC	Software Development Life Cycle
9.	HD	High-Level Design
10.	LLD	Low-Level Design

Product Scope

This software system will help students of computer science understand algorithms with the help of animations and run-time animation of new algorithms. The software will be designed to be more interactive and user-friendly so that students can grasp the working of algorithms. By enabling them to write their algorithm and see the performance and working of it at run time. Teachers can also use this system because it would give them a better medium to teach algorithms rather than relying on PowerPoint slides.

References

- "Algorithm Visualizer" by Barnini Goswami, Anushka Dhar, Akash Gupta, Antriksh Gupta, Volume 3 Issue 03 March 2021, International Research Journal of Modernization in Engineering Technology and Science, retrieved on 1 sept 2022
- https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects
  retrieved on 1st sept 2022
- https://theses.cz/id/3w4s3a/Mykhailo-Klunko-bc-thesis.pdfretrieved on 2nd sept 2022
- "Visualizing sorting algorithms" by Brian Faria, Rhode Island College, 2017. Retrieved on 2nd sept 2022.
- "ViSA: Visualization of Sorting Algorithms" by Tihomir Orehovački, ResearchGate, May 2012, retrieved on 10th sept 2022.
- IEEE Recommended Practice for Software Requirements Specifications, retrieved 29 Aug 2022
  from https://ieeexplore.ieee.org/document/278253
- Open proj, retrieved on 5th sept 2022
- Lucid chart, retrieved on 5th sept 2022
- Creatly, retrieved on 5th sept 2022

3. Overall Description

Product Perspective

The central perspective of this project is to help beginners to be able to visualize the basic algorithms and get a better understanding of the underlying operations. And obviously, it is needless to say that anyone willing to contribute is voted to use their creativity to make the visualizations even better and more attractive. One can add new Algorithms and visualization of their choice too.

• This project is for educational purposes.

• Extract the required feature and classify it among others.

Product Functions

The product should be able to perform the following operations:

- The product should work on different arrays and sorting algorithms (InsertionSort, Bubble Sort, Selection Sort).
- User should be able to restart or pause the visualization.
- It should be able to visualize each change made in the given array step by stepin a playful way.
- User must be able to step back or forward during execution.
- It may also have the functionality of generating a random array of randomsize.
- Indexes of the array should be shown throughout the execution.

User Classes and Characteristics

The goal is to design a web app to understand the sorting algorithms in a playful way that will be beneficial for the following type of users :

- **Students:** It will act as an e-learning platform for students to understand the conceptof sorting algorithms. It will help the students who cannot understand the algorithm from code or pseudo code. This visualizer will act as a dry run which will help the student to visualize the algorithm and easily code the Sorting Algorithms.

- **Teaching faculty**: It will help the Faculty teach students innovatively.
  Rather than totally relying on the PowerPoint presentation and whiteboards, the learning is dull. So, going through these types of visualization makes the teaching better, and the student will be able to grasp more.

Operating Environment

Since Java is cross-platform, you may use the application within the most popular
PC operating systems where Java is supported. Here are given minimal system
requirements for several operating systems.
Requirements:

- Operating system: minimum Windows XP SP3 or Mac OS X 10.4.10 or Ubuntu 8.04
- Java: Java SE 8 with minimum update 40 (8u40) or update 51 (8u51) in case of Windows 10 operating system
- Processor: Dual-Core processor, 1.8 GHz
- Memory: 512MB of RAM (1GB recommended)

Design and Implementation Constraints.

The following list presents the constraints, assumptions, dependencies, or guidelines that are imposed upon implementation of the Sorting Algorithm Visualizer

- Only a few visuals can be supported on the display screen due to the compact formfactor.

- There is no need for memory.

- The product must have an intuitive user interface that is easy for everyone.

- The software should load quickly, and transaction processing shouldn't take more thanthree seconds.

- The product is available for laptops and PCs only.

User Documentation

- User Manual
- Implementation Manual

4. External Interface Requirements

User Interfaces

The user interface will provide a good look and effects to make it more user-friendly. Teachers and students can operate the system very efficiently.

Hardware Interfaces

- Processor 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 1.38 GHz
- Installed RAM8.00 GB (7.73 GB usable)
- System type 64-bit operating system, x64-based processor
- EditionWindows 11 Home Single Language
- Version 21H2
- OS build 22000.856
- Experience Windows Feature Experience Pack 1000.22000.856.0
- Memory 512GB (SSD)

Software Interfaces

This software uses object-oriented and functional programming paradigms to organize the code. The use of HTML5 (Hypertext Markup Language 5), JavaScript, and CSS combine to form this project's implementation (Cascading Style Sheets). Our web app can run on operating systems like Ubuntu and some web browsers, like Mozilla Firefox, Google Chrome, and Safari integration.

5. System Features

Functional Requirements

Our product will provide a user-friendly interface for the user. The user can watch the animation of the array being sorted using three different sorting algorithms. The user has the advantage of changing the elements of the collection as well as the size of the collection; to rewind the step, there is one step back button provided on the interface.

Functional Requirements	**Function**
F. Req. 1	User-friendly Interface, where users can select the sorting they want to perform
F. Req. 2	After selecting the type of sort, a user can input the size of an array accordingly, or there will be a random generation of array elements.
F. Req. 3	Clicking on the run button starts executing the algorithm, and how sorting is done is shown animatedly.
F. Req. 4	There is a functionality to step forward or backward while an array is in execution.

Visualization of algorithms

Learning algorithms through visualizations can help students grasp algorithms better. Excellent and basic knowledge of DSA is essential to becoming a good developer. Algo Assist ensures to provides clear visualization and helps students understand algorithms.

6. Other Non-functional Requirements

Performance Requirements

Performance requirements for Algorithm Visualizer can be categorized into three. They are as follows:

- Response Time:

  It depends upon the time complexity of the algorithm you have chosen.
- Capacity:

  The application is standalone. There is no capacity requirement.
- Resource utilization:
  - 32-bit dual-core 2Ghz CPU with SSE2 support.
  - 2 GB RAM
  - OpenGL 2.1 compatible graphics card with 512 MB RAM

Safety And Security Requirements

Because this is an open web app available for free to everyone. There are no login credentials required to access our web app. So your credentials will remain secure with you.

Reliability:

As the application is standalone and does not depend on any server or database, there is no concern about keeping that kind of component up and running all the time. This indicates that there will not be any availability problems either. Accordingly, as long as there is no problem with the user's operating system, it is expected that the program will go on processing smoothly.

Usability:

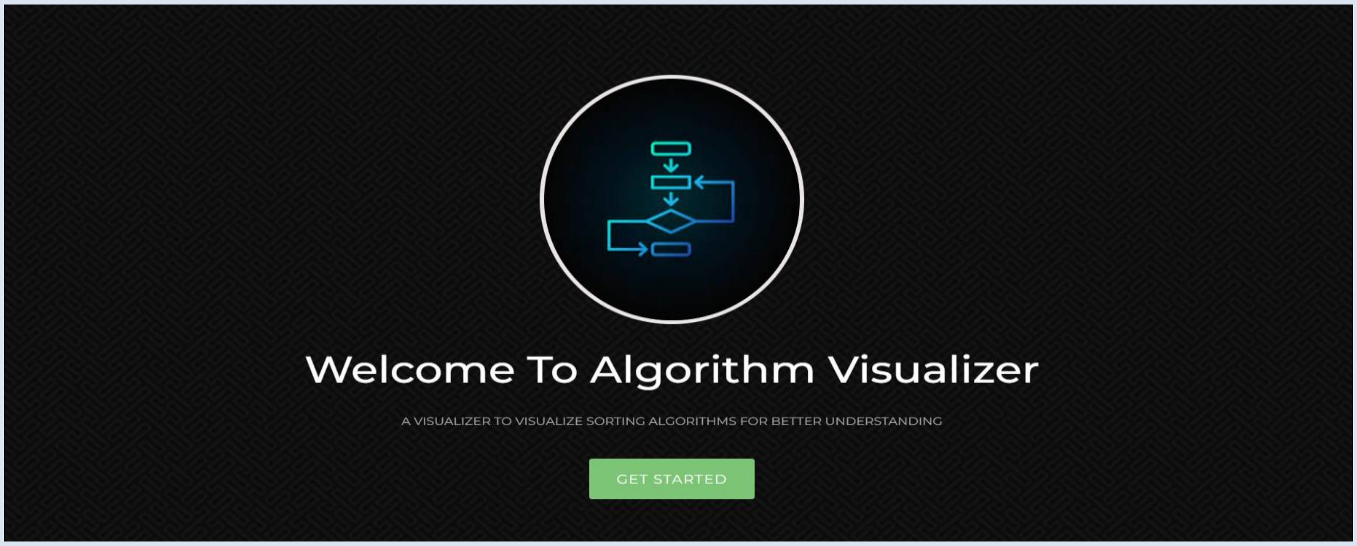
The user does not need to tackle any problems with any problems regarding the GUI as itaims to be simple but functional. The time for providing the result is sought to be kept minimum.

Business Rules

Anyone can use this application without considering any minimum requirement

2.5 User Story Cards

User Story Card : Front Of Card


Welcome To Algorithm Visualizer

A VISUALIZER TO VISUALIZE SORTING ALGORITHMS FOR BETTER UNDERSTANDING

User Story Card: Back side of Card

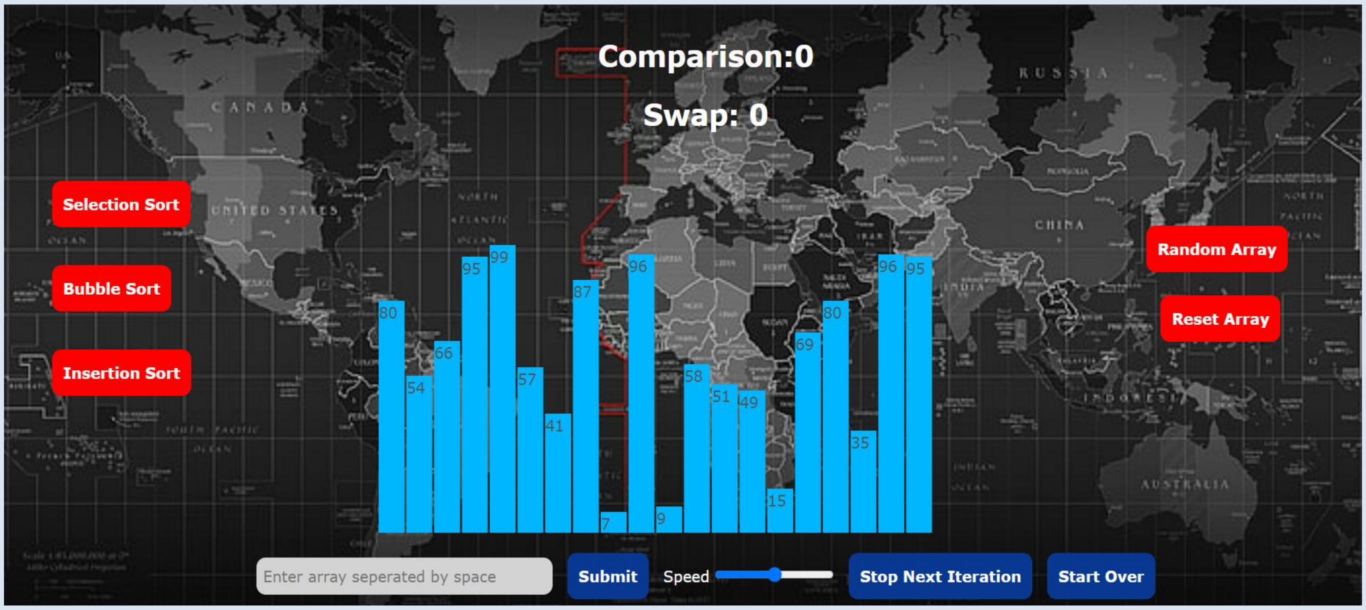
Success:

a) Interactive homepage with time delay.
b) User-Friendly
c) Navbar added to minimize scrolling effort.

Failure:

a) Heavy web app may take a longer time to load on Netlify.

User Story Card : Front Of Card



User Story Card: Back side of Card

Success:

a) Execution speed can be adjusted according to need.
b) User can reset the array.
c) Random array can be generated in the range 1-100 of size 20.
d) User can pause/resume the iteration process.
e) Users can get information of any particular sort with code/complexities.

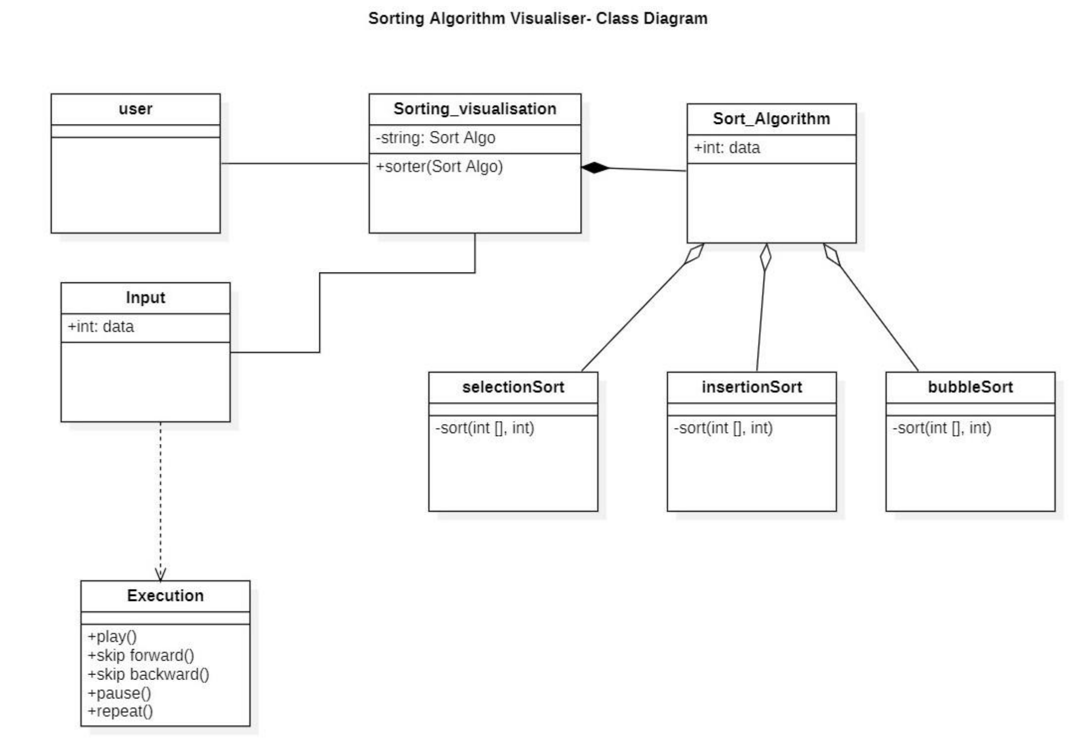
Failure:

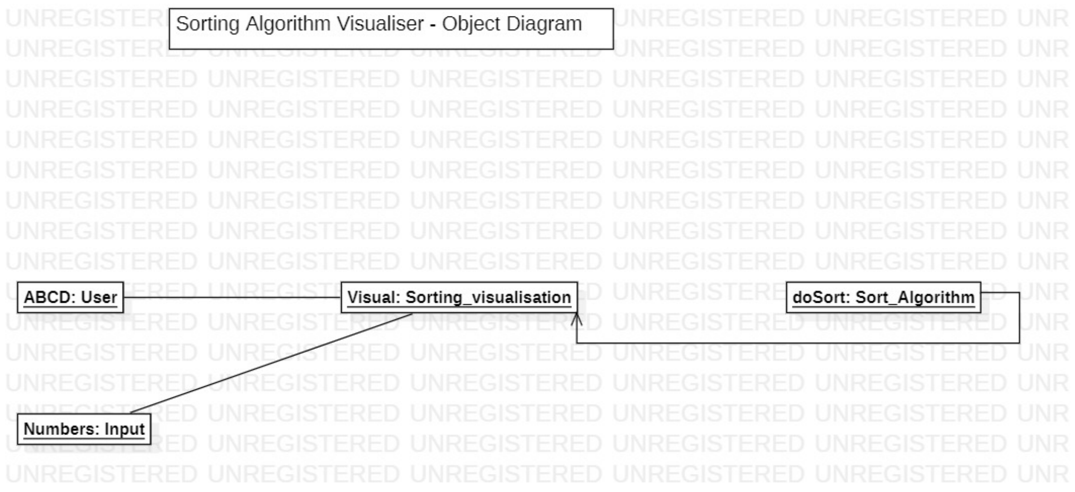
a) Less responsive on other devices (like mobile phones).
b) Skip forward/backward option is not included.

Constraints:

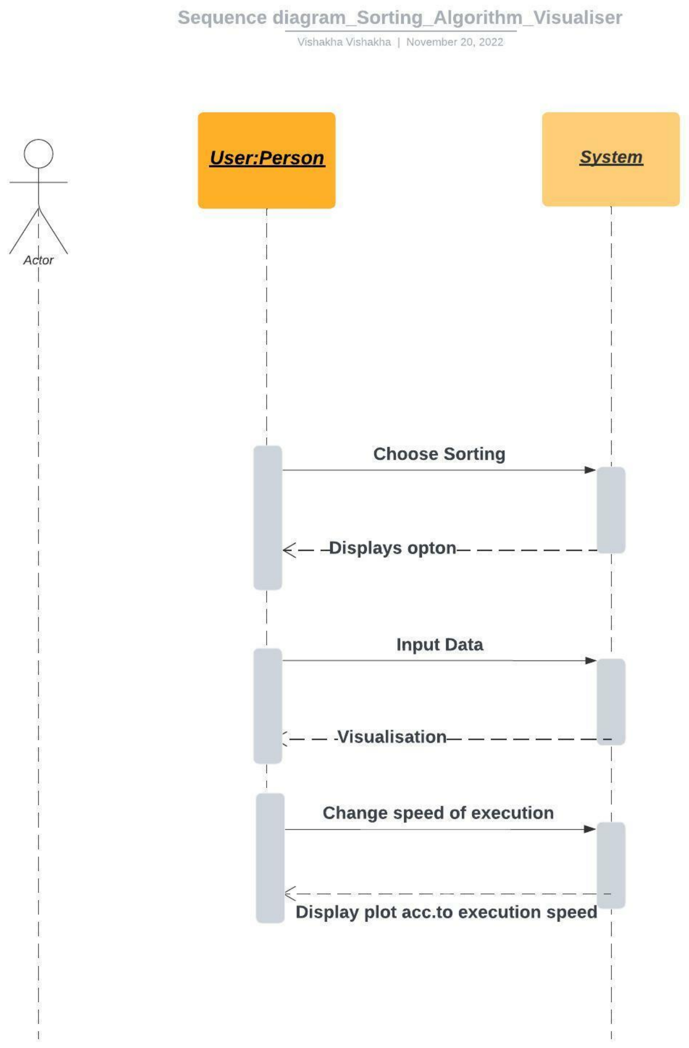
a) Random array generates an array of size 20 only.
b) User cannot watch the respective pseudo code along with visualization.

3.1 Class diagram and Object diagram




Sorting Algorithm Visualiser - Object Diagram
ABCD: User
Visual: Sorting_visualisation
doSort: Sort_Algorithm
Numbers: Input

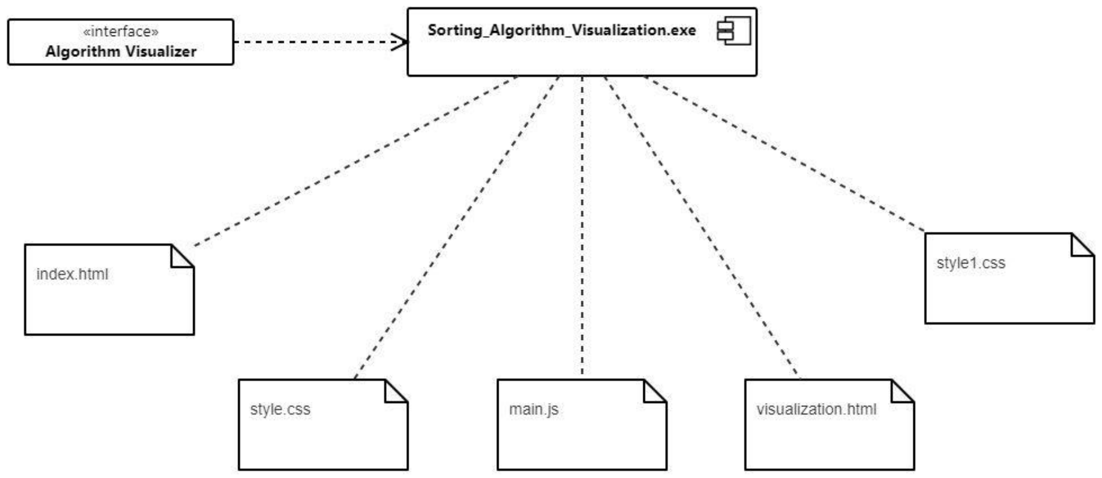
3.2 Sequence Diagram


Sequence diagram_Sorting_Algorithm_Visualiser
Vishakha Vishakha | November 20, 2022
Actor
User:Person
System
Choose Sorting
Displays opton
Input Data
Visualisation
Change speed of execution
Display plot acc.to execution speed

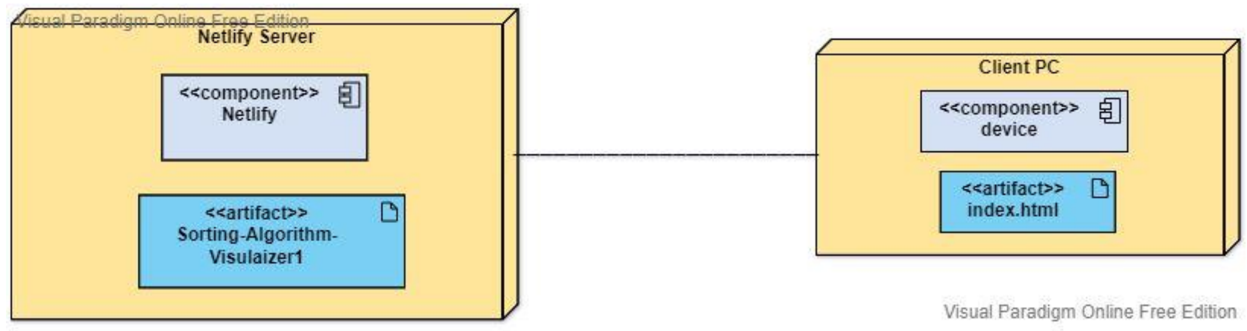
3.3 Database Design:ER Diagram

As there is no database in our project and only one entity i.e. user is there. So ,ER diagram is not feasible to make.

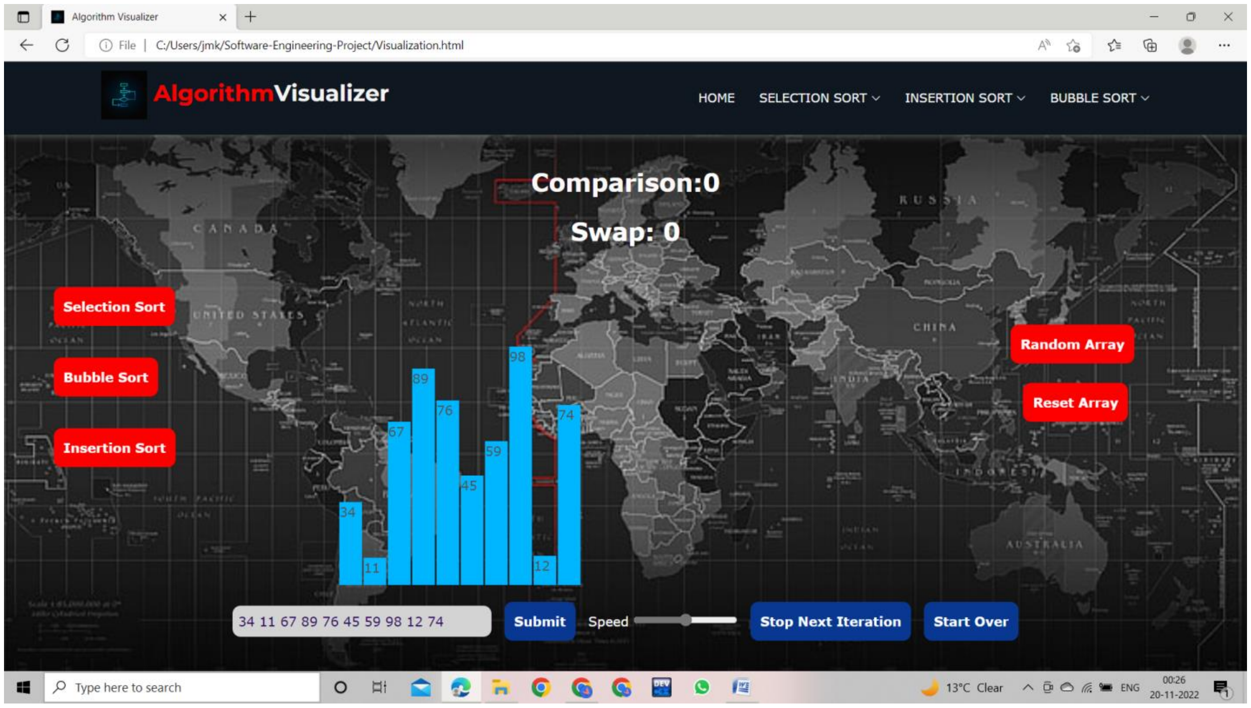
4.1 Component Diagram

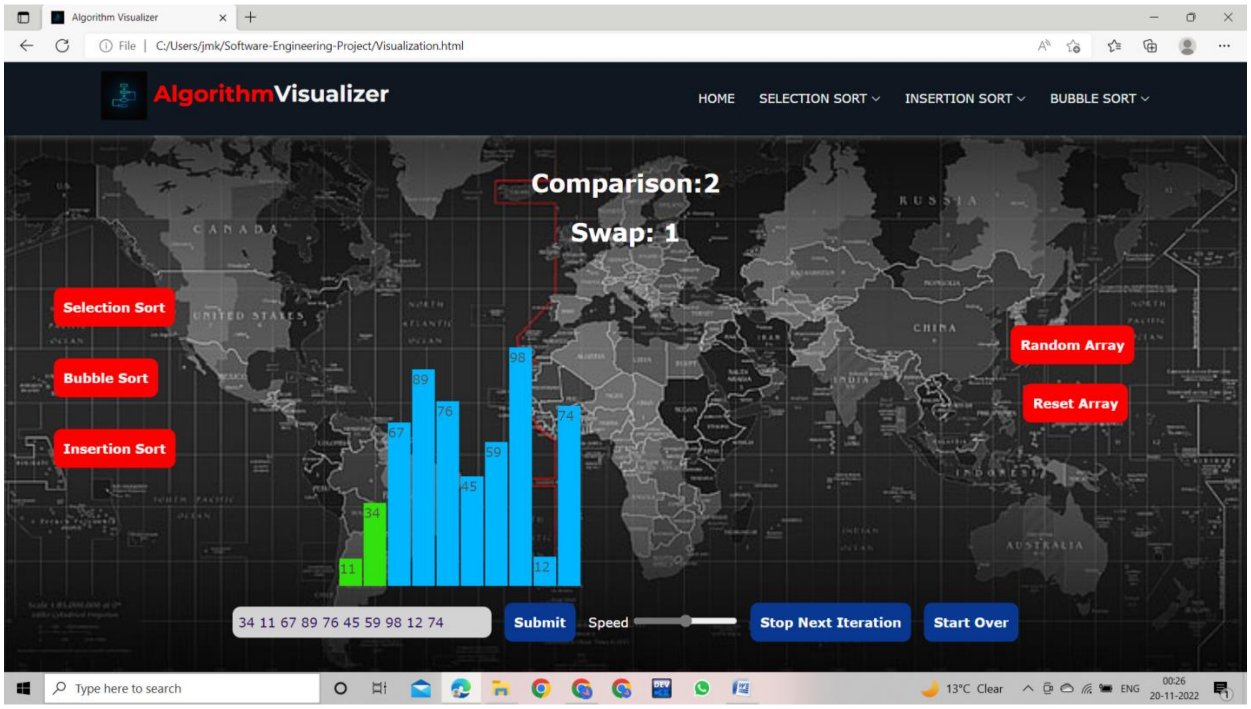


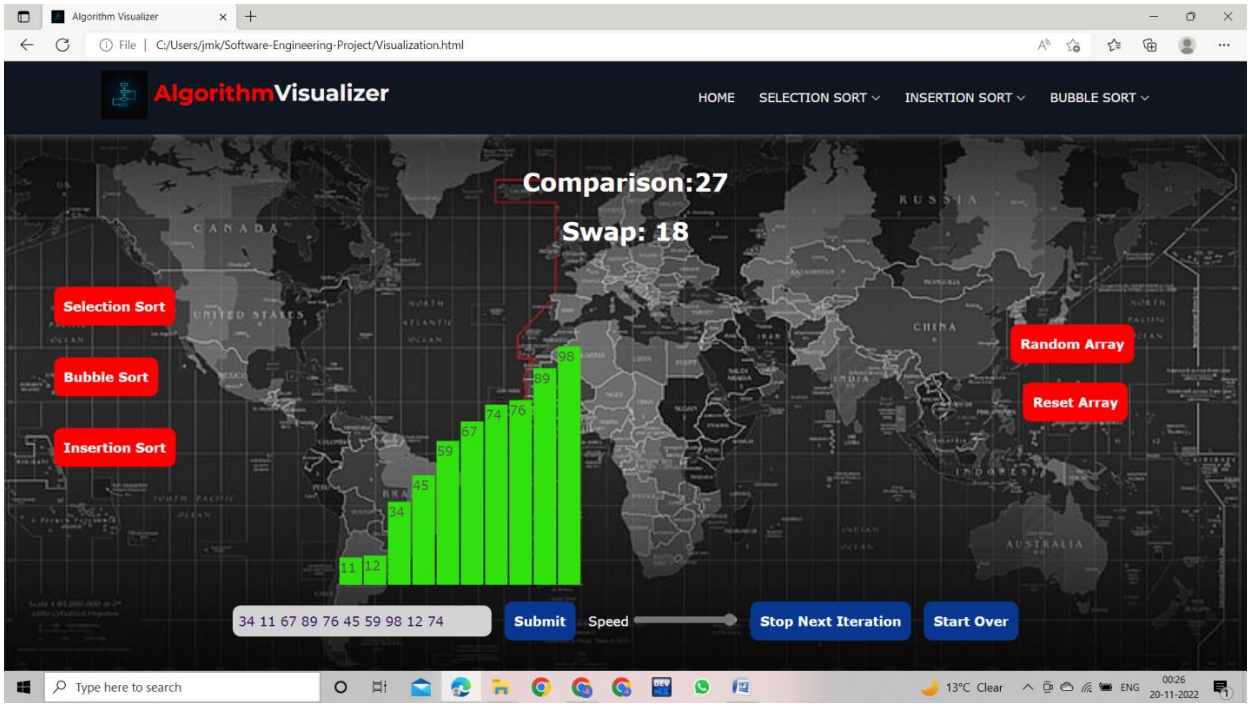
4.2 Deployment Diagram


Visual Paradigm Online Free Edition
Netlify Server
<<component>>
Netlify
<<artifact>>
Sorting-Algorithm-Visulaizer1
Client PC
<<component>>
device
<<artifact>>
index.html
Visual Paradigm Online Free Edition

4.3 Screenshot of Working Project







5.1 Cyclomatic Complexity(All modules)

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

Cyclomatic Complexity of our project is O(4).

5.2 Test Cases

Test Case #: 1	**Test Case Name:** Random Input Array	**Page: 1 of 1**
System: Laptop/PC	**Subsystem:** None	
Designed by: Neeraj,Vishakha,Ganesh	**Design Date:** 1/11/2022	
Executed by: Gaurav Gupta	**Execution Date:**16/11/2022	
Short Description: Test the Working of Visualiser		

Pre-Conditions:
User must have the basic information of what is meant by sorting.
The User has access to the Internet

Step	Action	Expected System Response	Pass/Fail
1	Click on Random Array	Array is generated in the form of bar plots of size 20.	Pass
2	Choose the Sorting	Animation begins showing the execution step-wise	Pass
3	Speed Slider	Changes the speed of Execution	Pass
4	Click Stop/Start-Over Button	Stops the execution and resume it	Pass as Expected
5	Click Reset Array Button	Reset The Array and Pages gets reloaded	Pass

Post-Conditions:
User gets the Final Sorted Array at the end.

Test Case #:2	**Test Case Name:** Positive Input Array	**Page: 1 of 1**
System: Laptop/PC	**Subsystem:** None	
Designed by: Neeraj , Vishakha , Ganesh	**Design Date:** 1/11/2022	
Executed by: Darpan Mittal	**Execution Date:**16/11/2022	
Short Description: Test the Working of Visualiser		

Pre-Conditions:
The User has access to the Internet

Step	Action	Expected System Response	Pass/Fail
1	Enter an own Arrayof Positve Numbers	Array is generated in the form of bar plots according to the input	Pass
2	Choose the Sorting	Execution Begins according to choosen sorting.	Pass
3	Speed Slider	Changes the speed of Execution	Pass
4	Click Stop/Start-Over Button	Stops the execution and resume it	Pass as Expected
5	Click Reset Array Button	Reset The Array and Pages gets reloaded	Pass

Post-Conditions:
User gets the Final Sorted Array at the end and got understood how it works.

Test Case #: 3	**Test Case Name:** Signed Input Array	**Page:** 1 of 1
System: Laptop/PC	**Subsystem:** None	
Designed by: Neeraj , Vishakha , Ganesh	**Design Date:** 1/11/2022	
Executed by: Anmol Assi	**Execution Date:**16/11/2022	
Short Description: Test the Working of Visualiser		

Pre-Conditions:
User must have the basic information of what is meant by sorting.
The User has access to the Internet

Step	Action	Expected System Response	Pass/Fail
1	Input the Array containing both Positive and Negative Numbers.	Bar Plots are formed containing both Positive and Negative numbers.	Pass
2	Choose the Sorting	Animation begins showing the execution step-wise	Pass
3	Speed Slider	Manages the speed of Execution according to the need.	Pass
4	Click Stop/Start-Over Button	Stops the execution and resume it	Pass as Expected
5	Click Reset Array Button	Delete the array and reloads the page	Pass

Post-Conditions:
User gets the Final Sorted Array at the end.

Test Case #: 4	**Test Case Name:** Large Input Array **Page: 1 of 1**
System: Laptop/PC	**Subsystem:** None
Designed by: Neeraj , Vishakha , Ganesh	**Design Date:** 1/11/2022
Executed by: Amit Kumar	**Execution Date:**16/11/2022
Short Description: Test the Working of Visualiser	

Pre-Conditions:
User must have the basic information of what is meant by sorting.
The User has access to the Internet

Step	Action	Expected System Response	Pass/Fail
1	Input the own Array of larger size say (100)	Bar Plots are formed, but only showing first 35 numbers only.	Fail
2	Choose the Sorting	Execution started according to the chosen sort	Pass
3	Speed Slider	Increases or Decreases the speed of execution.	Pass
4	Click Stop/Start-Over Button	Pause the execution and resume it.	Pass as Expected
5	Click Reset Array Button	Reloads the page and Input gets deleted.	Pass

Post-Conditions:
User gets the Final Sorted Array at the end.